مقاله‌ای خلاصه در مورد پایگاه‌داده sqlite3 و python3

فهرست

وبلاگ من : amirkouhkan.ir

بهتره اول یه مقدمه درباره این پایگاه‌داده بگم :

دکتر ریچارد هیپ در سال ۲۰۰۰ در زمانی که از طریق جنرال داینامیکس با نیروی دریایی ایالات متحده امریکا کار می‌کرد طراحی کرد. او در آن زمان مشغول کار روی برنامه‌های مربوط به ناوشکن هایی بود که به موشک‌های هدایت‌شونده مجهز بودند و تا آن زمان از پایگاه داده آی‌بی‌ام اینفورمیکس استفاده می‌کردند. هدف از ساخت اس‌کیوال لایت این بود که این برنامه‌ها بتوانند بدون نصب یا مدیریت پایگاه داده مستقل اجرا شوند. نسخهٔ اول نرم‌افزار در اوت ۲۰۰۰ انتشار یافت. در نسخهٔ ۲٫۰ ساختار داخلی اس‌کیوال لایت تغییر یافت و از یک درخت بی در آن استفاده گردید. در نسخه ۳٫۰ که قسمتی از هزینه‌هایش توسط ای‌اوال تامین شد، پشتیبانی چندزبانی و چند تغییر بزرگ دیگر در اس‌کیوال لایت رخ داد.

بطور خیلی خلاصه میخوام این مقاله رو با ساختن یک برنامه‌ای که نام و نام‌خانوادگی و شماره‌تلفن افراد را دریافت میکند و ذخیره میکند.

امکانات برنامه :

۱. وارد کردن اطلاعات
۲. ویرایش اطلاعات
۳. حذف کردن اطلاعات
۴. نمایش دادن اطلاعات

این برنامه رو میخوایم با زبان برنامه‌نویسی پایتون ۳ و پایگاه‌داده‌ی sqlite3 بسازیم :)

اولین کاری که باید بکنیم این است که کتابخانه sqlite3 را به درون برنامه import کنیم.

```
import sqlite3
```

حالا باید یک پایگاه‌داده بسازیم، دستور کلی ساخت پایگاه‌داده :

```
[database variable] = sqlite3.connect('[database name].db')
```

database variable نام متغیری است که ما در طول برنامه کارهای اضافه کردن و حذف کردن و... را بوسیله این متغیر انجام میدهیم.

database name هم نامی‌است که برای پایگاه‌داده خود در نظر میگیریم.

حالا ما نام متغیر db و نام دیتابیس data را در نظر میگیریم.

من در این مقاله هر کدام از اعمال را درون یک تابع مینویسم! شما هر جور که راحت هستید انجام دهید.

کد های ما تا بدینجا :

```
import sqlite3

def createDatabase():
        global db
        db = sqlite3.connect('data.db')
        print(“Created Successfully :D “)
        db.close()
```

حالا باید یک جدول بسازیم تا اطلاعات خودمون رو داخلش قرار بدیم. جدول ما شامل یک ID هست تا کار های حذف و ویرایش آسون بشه!

شکل کلی تعریف جدول :

```
[database].execute('create table [tbl-name] (field type,field type),... ')
```

و اما جدولی که لازم داریم :

```
def createTable():
	createDatabase()
	db.execute('create table data (id int, name text, family text,
phoneNumber text)')
	print(“table is created”)
	db.close()
```

نکته اول : تنها یک بار باید تابع createTable() رو باید فراخوانی کنیم!

نکته دوم : باید تابع createDatabase() را در هر تابع‌ای که میخواهیم عملی انجام دهد را فراخوانی کنیم تا بتوانیم به پایگاه‌داده متصل شویم.

نکته سوم : در پایان دستورات هر کدام از توابعمان پایگاه‌داده را ببندیم.

حالا عمل وارد کردن داده به پایگاه‌داده را بیاموزیم

جستار(query) کلی وارد کردن داده(Data) به پایگاه داده به شکل زیر است :

```
[database].execute('insert into [table] (field,field,...) values(?,?,...),
(value,value,...)
```

تابع وارد کردن id , name , family , phoneNumber :

```
def insertDatabase():
        createDatabase()
        db.execute('insert into
data(id,name,family,phoneNumber)values(?,?,?,?)',
(1,”Amir”,”Kouhkan”,”09000000000”)
        db.execute('insert into
data(id,name,family,phoneNumber)values(?,?,?,?)',
(2,”Sajad”,”Atashkar”,”09000000000”)
        db.commit()
        print(“Inserted seccessfully”)
        db.close()
```

در این تابع :

- ✔ ابتدا تابع createDatabase() را فراخوانی کردیم تا اتصال به پایگاه‌داده را داشته باشیم
- ✔ سپس ۲ جستار مربوط به وارد کردن value به پایگاه‌داده را نوشتیم
- ✔ با استفاده از تابع commit() اطلاعات را به پایگاه‌داده منتقل میکنیم
- ✔ و بعد از آن یک پیام موفقیت ارسال میکنیم
- ✔ در نهایت اتصال به پایگاه‌داده را قطع میکنیم

اما این شکل از نوشتن برنامه یه نوع دیکتاتوریه! چون تنها این امکان رو به ما میده که value هایی که خودمون تعریف کردیم رو به پایگاه‌داده انتقال بده! اما اگر بخوایم برنامه ما تعاملی (interactive) باشه میتونیم تابع خود را به شکل زیر ویرایش کنیم :

```
def insertDatabase():
        createDatabase()
        ID = int(input(“Enter an ID : “))
        name = input(“Enter a name : “)
        family = input(“Enter a family : “)
        phoneNumber= input(“Enter a phonenumber : “)
        db.execute('insret into data(id,name,family,phoneNumber)values(?,?,?,?)'
,(ID,name,family,phoneNumber))
        db.commit()
        print(“The operation is successfully”)
        db.close()
```

حالا خوبی‌های این نوع تعریف کردن تابع این‌است‌که مجبور نیستیم برای وارد کردن یه‌سری اطلاعات جدید کدمنبع خود را ویرایش کنیم!

حالا فرض رو بر این میزارم که میخوایم یه برنامه تعاملی بنویسیم!

نوشتن تابع حذف کردن

جستار کلی مربوط به حذف کردن اطلاعات :

```
[database].execute('delete from [table] where [condition]=[value]')
```

حالا جستار مربوط به برنامه خودمون رو مینویسیم:

```
def deleteDatabase():
        createDatabase()
        ID = input(“Enter an ID : “)
        db.execute('delete from data where id ={0}'.format(ID))
        db.commit()
        print(“Deleted Successfully”)
        db.close()
```

نکات مربوط به این تابع :

- ما برای حذف کردن یک دسته از اطلاعات وارد شده باید از یک داده ثابت استفاده کنیم که معمولاً اون داده id هست
- یک id دریافت میکنیم که با این id داده‌ها را پاک میکنیم
- برای اینکه برنامه‌ی ما انعطاف پذیر باشه از جانگهدارها استفاده می‌کنیم {0}

تابع ویرایش اطلاعات

ویرایش اطلاعات را هم بر اساس یک id انجام میدهیم.

جستار کلی مربوط به ویرایش اطلاعات :

```
[database].execute('update data set field=value,feild=value where condition=value')
```

جستار برنامه ما و تابع مربوط به ویرایش به شرح زیر است :

```
def updateDatabase():
        createDatabase()
        ID = int(input(“Enter an ID :”)
        name = input(“Enter a new name : “)
        family = input(“Enter a new family : “ )
        phoneNumber = input(“Enter a new phonenumber : “)
        db.execute('update data set name=”{1}”,family=”{2}”,phoneNumber=”{3}” where id={0}'.format(ID,name,family,phoneNumber)
        db.commit()
        print(“Edited Successfully”)
        db.close()
```

نکات مربوط به این تابع :

- برای ویرایش، id و سایر اطلاعات را از کاربر دریافت میکنیم.
- مهمترین نکته‌ای که وجود داره اینه‌که name,family,phoneNumber رشته هستند و باید جانگهدار آن‌ها درون ” ” قرار بگیره، اما id بخاطر این که از نوع int هست نیاز نیست درون ” ” قرار بگیره!

نمایش اطلاعات

و آخرین کاری که انجام میدیم اینه که اطلاعات رو نمایش بدیم. راه‌های زیادی هست اما من سریع‌ترین راه رو مینویسم

جستار کلی نمایش اطلاعات :

```
[database].execute('select * from [table]')
```

حالا جستار مربوط به تابع خودمان رو مینویسیم :

```
def showDatabase():
	createDatabase()
	getData = db.execute('select * from data')
	for show in getData:
		print(show)
	db.close()
```

نکات مربوط به این تابع :

- علامت * یعنی تمام داده‌ها
- ما درون یک متغیر اطلاعات پایگاه‌داده را قرار میدهیم و سپس با استفاده از حلقه به نمایش میگذاریم

استفاده از توابع

حالا توابع رو درون تابع اصلی استفاده میکنیم :

```
def main():
	print(“List of jobs : \n1. Create Database \n2. Create Table \n3. Insert \n4.
Delete \n5. Edit \n6. show\n7. Exit \n”)
	getID = int(input(“Enter an ID : “))
	while getID < 8 :
		if getID == 1:
			createDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 2:
			createTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 3:
			insertDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 4:
			deleteTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 5:
			updateTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 6:
			showDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 7:
			break
		else:
			getID = int(input(“Error! Enter an ID, again : “)
```

تمام کد ها :

```
#!/usr/bin/python3
import sqlite3

def createDatabase():
	global db
	db = sqlite3.connect('data.db')
	print(“Created Successfully :D “)
	db.close()
def createTable():
	createDatabase()
	db.execute('create table data (id int, name text, family text,
phoneNumber text)')
	print(“table is created”)
	db.close()
def insertDatabase():
	createDatabase()
	ID = int(input(“Enter an ID : “))
	name = input(“Enter a name : “)
	family = input(“Enter a family : “)
	phoneNumber= input(“Enter a phonenumber : “)
	db.execute('insret into data(id,name,family,phoneNumber)values(?,?,?,?)'
,(ID,name,family,phoneNumber))
	db.commit()
	print(“The operation is successfully”)
	db.close()
def deleteDatabase():
	createDatabase()
	ID = input(“Enter an ID : “)
	db.execute('delete from data where id ={0}'.format(ID))
	db.commit()
	print(“Deleted Successfully”)
	db.close()
def updateDatabase():
	createDatabase()
	ID = int(input(“Enter an ID :”)
	name = input(“Enter a new name : “)
	family = input(“Enter a new family : “ )
	phoneNumber = input(“Enter a new phonenumber : “)
	db.execute('update data set
name=”{1}”,family=”{2}”,phoneNumber=”{3}” where
id={0}'.format(ID,name,family,phoneNumber)
	db.commit()
	print(“Edited Successfully”)
	db.close()
def showDatabase():
	createDatabase()
	getData = db.execute('select * from data')
	for show in getData:
		print(show)
	db.close()
```

```
def main():
	print(“List of jobs : \n1. Create Database \n2. Create Table \n3. Insert \n4.
Delete \n5. Edit \n6. show\n7. Exit \n”)
	getID = int(input(“Enter an ID : “))
	while getID < 8 :
		if getID == 1:
			createDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 2:
			createTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 3:
			insertDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 4:
			deleteTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 5:
			updateTable()
			getID = int(input(“Enter an ID : “))
		elif getID == 6:
			showDatabase()
			getID = int(input(“Enter an ID : “))
		elif getID == 7:
			break
		else:
			getID = int(input(“Error! Enter an ID, again : “)
if __name__==”__main__”==main()
```